**Name:**  **Ahmad Sultan**

**Class:**  **BSCS**

**Semester:**  **5th**

**Ag-No:**  **2020-ag-6374**

**Subject**  **Artificial Intelligence**

**Submitted to:**  **Abdul Mannan**

**==========================================================**

**====================================================================================================================**

**Python Assignment #01**

**01) Variable:**

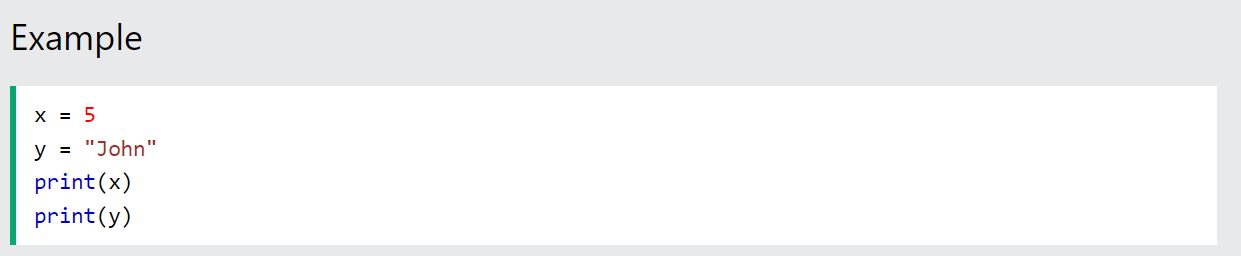
Variables are containers for storing data values.

## **Creating Variables:**

Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

**Example**



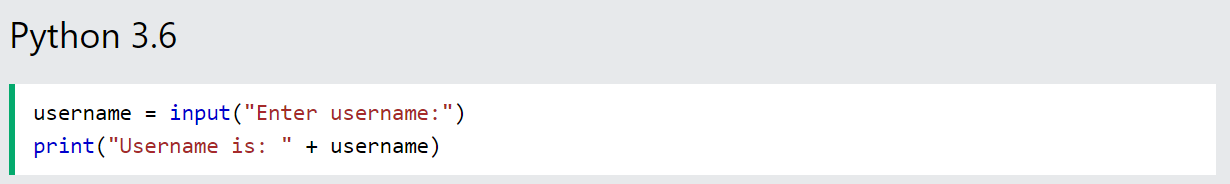
**02) Input:**

Python allows for user input.

That means we are able to ask the user for input.

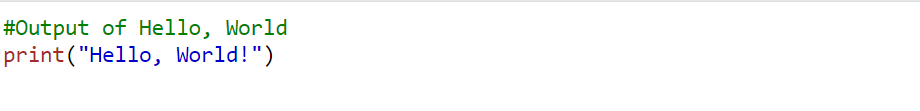
Python 3.6 uses the input() method.

**Example**



**03) Output**

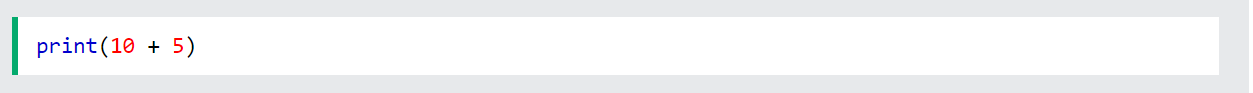
Python uses print() method for output.



**04) Operators**

Operators are used to perform operations on variables and values.

In the example below, we use the + operator to add together two values:

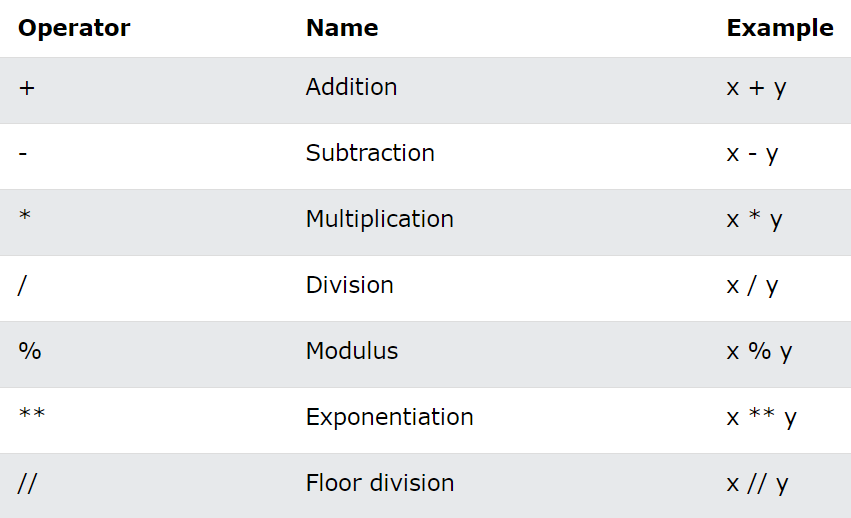


Python divides the operators in the following groups:

* Arithmetic operators
* Assignment operators
* Comparison operators
* Logical operators
* Identity operators
* Membership operators
* Bitwise operators

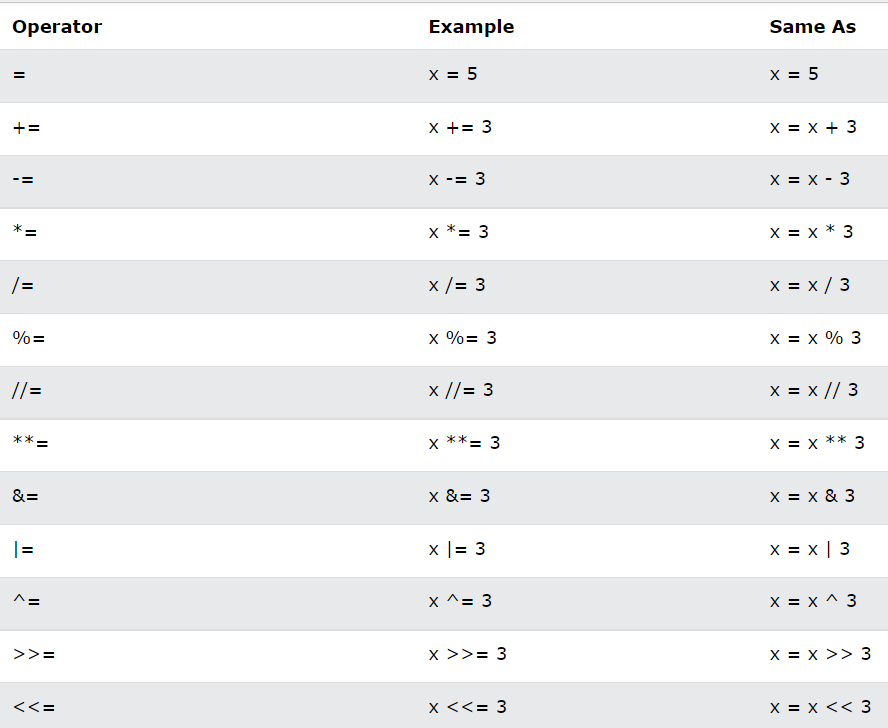
## **Python Arithmetic Operators**

Arithmetic operators are used with numeric values to perform common mathematical operations:



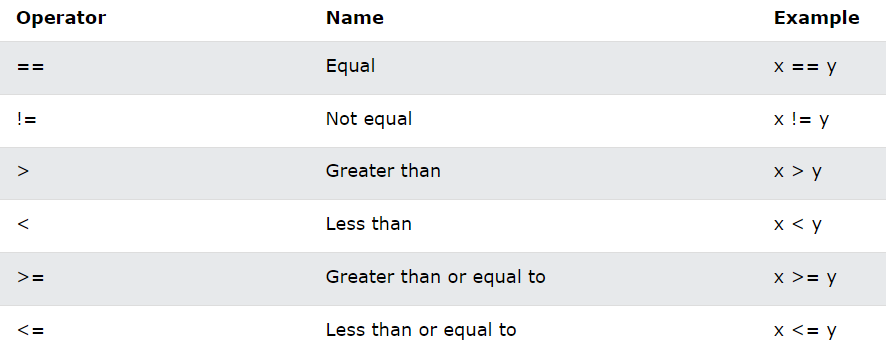
## **Python Assignment Operators**

Assignment operators are used to assign values to variables:



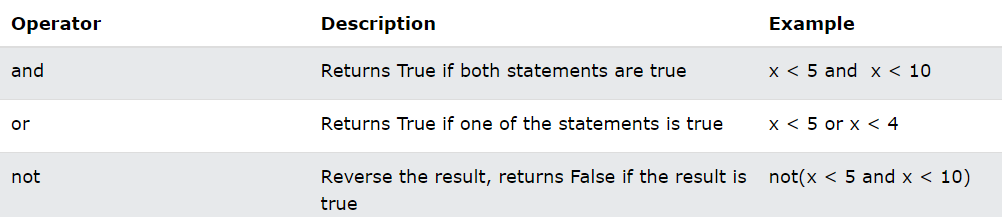
## **Python Comparison Operators**

Comparison operators are used to compare two values:



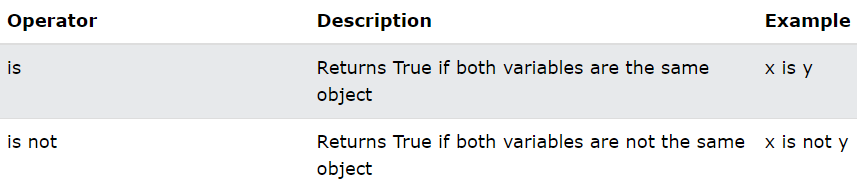
## **Python Logical Operators**

Logical operators are used to combine conditional statements:



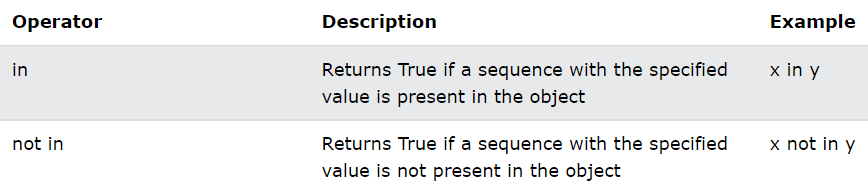
## **Python Identity Operators**

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:



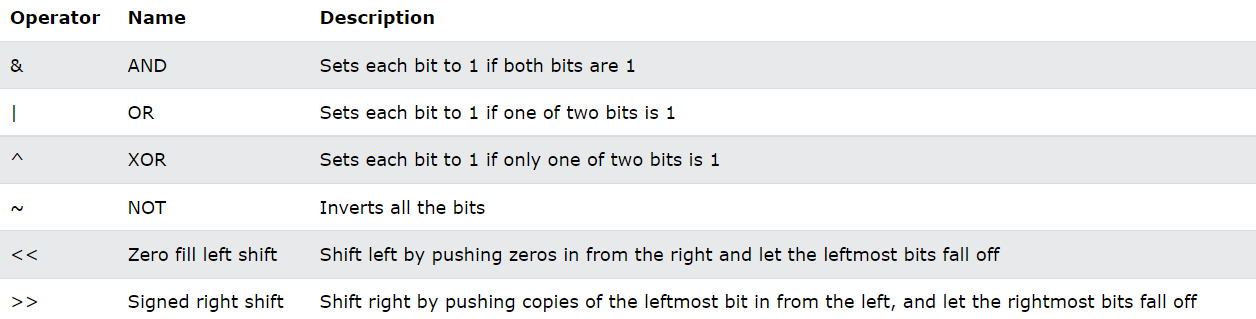
## **Python Membership Operators**

Membership operators are used to test if a sequence is presented in an object:



## **Python Bitwise Operators**

Bitwise operators are used to compare (binary) numbers:



**05) Control Structures**

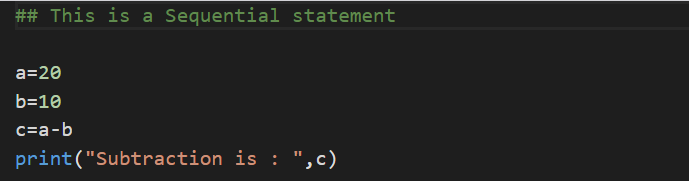
A program’s **control flow** is the order in which the program’s code executes.

Python has *three* types of control structures:

* + **Sequential** - default mode
  + **Selection** - used for decisions and branching
  + **Repetition** - used for looping, i.e., repeating a piece of code multiple times.

## **1. Sequential**

**Sequential statements** are a set of statements whose execution process happens in a sequence. The problem with sequential statements is that if the logic has broken in any one of the lines, then the complete source code execution will break.



## **2. Selection/Decision control statements**

In Python, the selection statements are also known as *Decision control*  *statements* or *branching statements*.

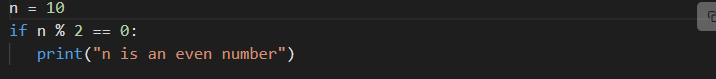
The selection statement allows a program to test several conditions and execute instructions based on which condition is true.

Some Decision Control Statements are:

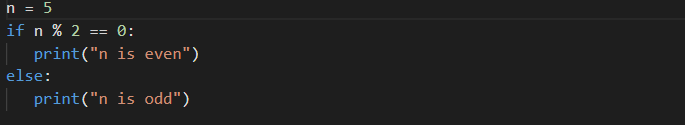
* + Simple if
  + if-else
  + nested if
  + if-elif-else

**Simple if:**

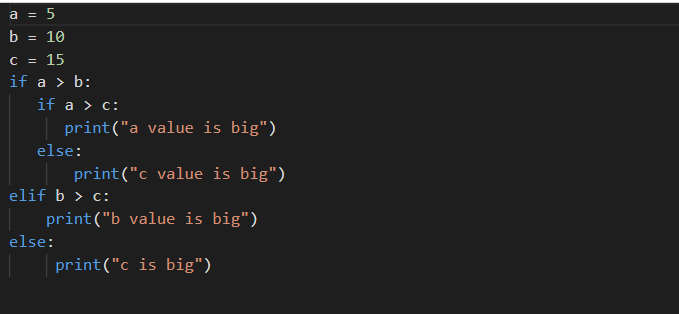
*If statements* are control flow statements that help us to run a particular code, but only when a certain condition is met or satisfied. A *simple*  *if* only has one condition to check.

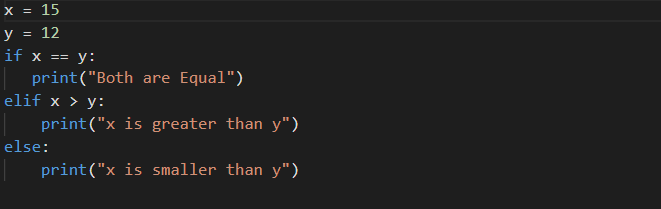


**if-else:** The *if-else statement* evaluates the condition and will execute the body of if if the test condition is True, but if the condition is False, then the body of else is executed.



**nested if:** *Nested if statements* are an if statement inside another if statement.



**if-elif-else:** The *if-elif-else statement* is used to conditionally execute a statement or a block of statements. 

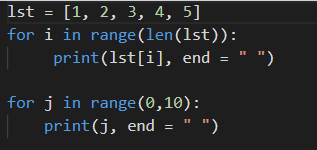
## **3. Repetition**

A **repetition statement** is used to repeat a group(block) of programming instructions.

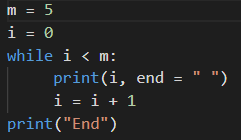
In Python, we generally have two loops/repetitive statements:

* + for loop
  + while loop

**for loop:** A *for loop* is used to iterate over a sequence that is either a list, tuple, dictionary, or a set. We can execute a set of statements once for each item in a list, tuple, or dictionary.



**while loop:** In Python, *while loops* are used to execute a block of statements repeatedly until a given condition is satisfied. Then, the expression is checked again and, if it is still true, the body is executed again. This continues until the expression becomes false.



**==============================================================================================================================================================================**